# Лекция 9

06 марта

# Обратная задача

```
void f(int *xp, int *yp, int *zp) {
        ???
}
```

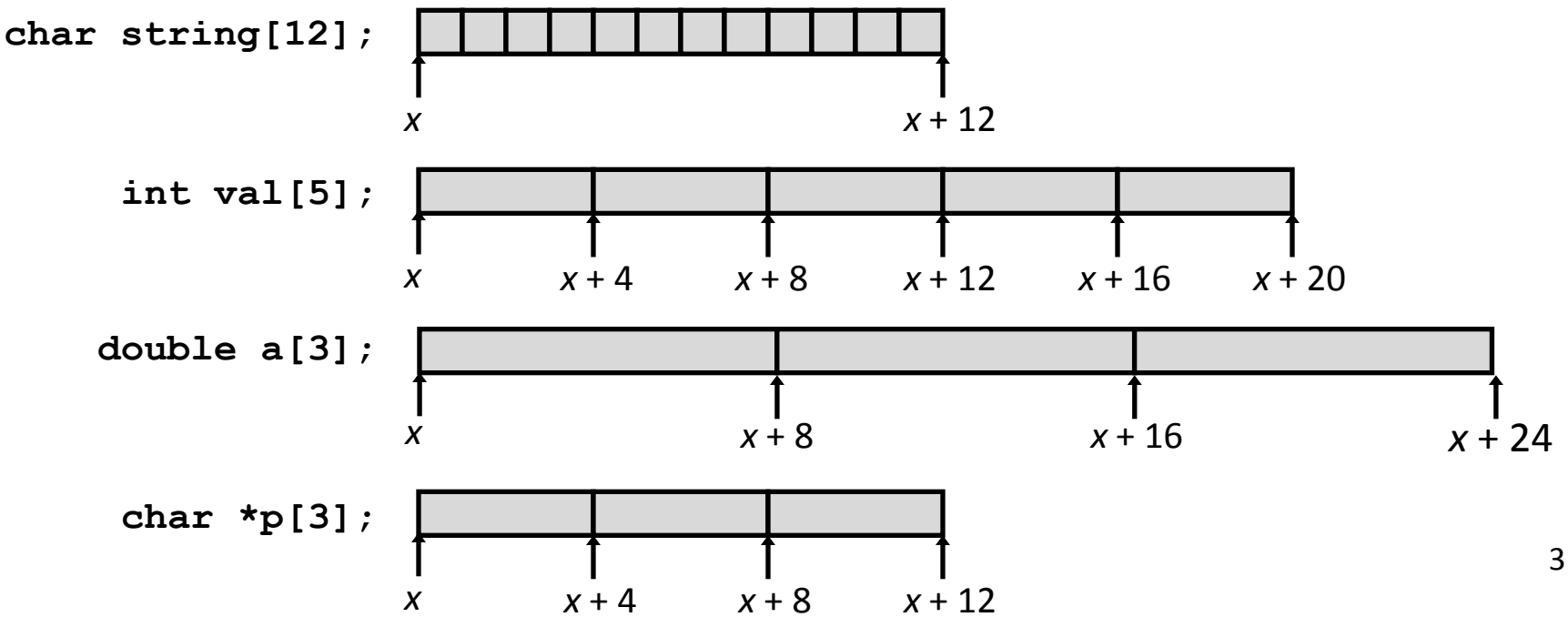
```
exchange:
   ... ; пролог функции
   mov edi, [ebp + 8]   ; (1)
   mov edx, [ebp + 12]  ; (2)
   mov ecx, [ebp + 16]  ; (3)
   mov ebx, [edx]       ; (4)
   mov esi, [ecx]       ; (5)
   mov eax, [edi]       ; (6)
   mov [edx], eax       ; (7)
   mov [ecx], ebx       ; (8)
   mov [edi], esi       ; (9)
   ... ; эпилог функции
```

| Параметр | Размещение |
|---|---|
| **xp** | **ebp + 8** |
| **yp** | **ebp + 12** |
| **zp** | **ebp + 16** |

# Массивы – размещение в памяти

**`T  A[L];`**

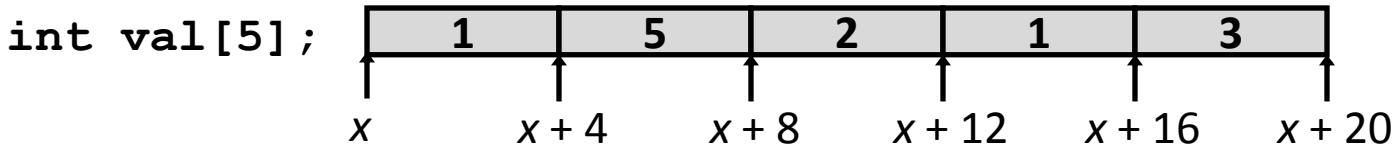
- Массив элементов типа T, размер массива – L
- Массив располагается в непрерывном блоке памяти размером *L* * **`sizeof`** (*T*) байт



# Доступ к элементам массива

**`T  A[L];`**

- Массив элементов типа T, размер массива – L
- Идентификатор A может использоваться как указатель на элемент массива с индексом 0. Тип указателя – T*




- Задачи …

| J*cc* | Условие | Описание |
|---|---|---|
| JE | ZF | Равно / Ноль |
| JNE | ~ZF | Не равно / Не ноль |
| JS | SF | Отрицательное число |
| JNS | ~SF | Неотрицательное число |
| JG | ~(SF^OF)&~ZF | Больше (знаковые числа) |
| JGE | ~(SF^OF) | Больше либо равно (знаковые числа) |
| JL | (SF^OF) | Меньше (знаковые числа) |
| JLE | (SF^OF)\|ZF | Меньше либо равно (знаковые числа) |
| JA | ~CF&~ZF | Больше (числа без знака) |
| JB | CF | Меньше (числа без знака) |

```c
int absdiff(int x, int y) {
    int result;
    if (x > y) {
      result = x-y;
    } else {
      result = y-x;
    }
    return result;
}
```

```
absdiff:
   push  ebp
   mov   ebp, esp
   mov   edx, dword [8 + ebp]  ; (1)
   mov   eax, dword [12 + ebp] ; (2)
   cmp   edx, eax              ; (3)
   jle   .L6                   ; (4)
   sub   edx, eax              ; (5)
   mov   eax, edx              ; (6)
   jmp   .L7                   ; (7)
.L6:                           ; (8)
   sub eax, edx                ; (9)
.L7:                           ; (10)
   pop ebp
   ret
```

```c
int goto_ad(int x, int y) {
  int result;
  if (x <= y) goto Else;
  result = x-y;
  goto Exit;
Else:
  result = y-x;
Exit:
  return result;
}
```

```asm
absdiff:
   push  ebp
   mov   ebp, esp
   mov   edx, dword [8 + ebp]  ; (1)
   mov   eax, dword [12 + ebp] ; (2)
   cmp   edx, eax              ; (3)
   jle   .L6                   ; (4)
   sub   edx, eax              ; (5)
   mov   eax, edx              ; (6)
   jmp   .L7                   ; (7)
.L6:                           ; (8)
   sub eax, edx                ; (9)
.L7:                           ; (10)
   pop ebp
   ret
```

# Условная передача данных

```
val = Test ? Then_Expr : Else_Expr;

val = x>y ? x-y : y-x;
```

```
   nt = !(Test);
   if (nt) goto Else;
   val = Then_Expr;
   goto Done;
Else:
   val = Else_Expr;
Done:
   ...
```

```
tmp_val = Then_Expr;
val = Else_Expr;
t = Test;
if (t) val = tmp_val;
```

```c
int absdiff(int x, int y) {
    int result;
    if (x > y) {
      result = x-y;
    } else {
      result = y-x;
    }
    return result;
}
```

| Регистр | Значение |
|---|---|
| **edi** | **x** |
| **esi** | **y** |

```
absdiff:
   ...
   mov edx, edi
   sub edx, esi      ; tmp_val:edx = x-y
   mov eax, esi
   sub eax, edi      ; result:eax = y-x
   cmp edi, esi      ; Compare x:y
   cmovg eax, edx    ; If >, result:eax = tmp_val:edx
   ...
```

# Оператор do-while

```
int pcount_do(unsigned x) {
  int result = 0;
  do {
    result += x & 0x1;
    x >>= 1;
  } while (x);
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

# Оператор do-while

| Регистр | Значение |
|---|---|
| **edx** | **x** |
| **ecx** | **result** |

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

```
   mov ecx, 0     ;   result = 0
.L2:              ; loop:
   mov eax, edx
   and eax, 1     ;   t = x & 1
   add ecx, eax   ;   result += t
   shr edx, 1     ;   x >>= 1
   jne .L2        ;   If !0, goto loop
```

# Оператор while

```
int pcount_while(unsigned x) {
  int result = 0;
  while (x) {
    result += x & 0x1;
    x >>= 1;
  }
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  if (!x) goto done;
  result += x & 0x1;
  x >>= 1;
  goto loop;
done:
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
  if (!x) goto done;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
done:
  return result;
}
```

# Оператор for

```
#define WSIZE 8*sizeof(int)

int pcount_for(unsigned x) {
  int i;
  int result = 0;
  for (i = 0; i < WSIZE; i++) {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  return result;
}
```

```
int pcount_for_gt(unsigned x) {
  int i;
  int result = 0;
  i = 0;
  if (!(i < WSIZE))
    goto done;
 loop:
  {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  i++;
  if (i < WSIZE)
    goto loop;
 done:
  return result;
}
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

| Регистр | Значение |
|---|---|
| **ecx** | **x** |
| **edx** | **p_pred** |
| **ebx** | **pred** |
| **eax** | **res** |

```asm
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

| Регистр | Значение |
|---|---|
| ecx | x |
| edx | p_pred |
| ebx | pred |
| eax | res |

```asm
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

| Регистр | Значение |
|---|---|
| ecx | x |
| edx | p_pred |
| ebx | pred |
| eax | res |

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

| Регистр | Значение |
|---|---|
| ecx | x |
| edx | p_pred |
| ebx | pred |
| eax | res |

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```